Distance-based adaptive k-neighborhood selection

Albrecht Zimmermann

Report CW 647, November 2013

KU Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Distance-based adaptive k-neighborhood selection

Albrecht Zimmermann

Report CW 647, November 2013

Department of Computer Science, KU Leuven

Abstract

The k-nearest neighbor classifier follows a simple, yet powerful algorithm: collect the k data points closest to an unlabeled instance, according to a given distance measure, and use them to predict that in- stance's label. The two components, the parameter k governing the size of used neighborhood, and the distance measure, essentially determine success or failure of the classifier. In this work, we propose to reverse the use of outlier-detection techniques that are based on k-neighborhoods in order to determine the value of k. To achieve this, we invert the workings of these techniques: instead of using a fixed k to decide whether an instance is an outlier, we stop growing the k-neighborhood as soon as the unlabeled instance would be given outlier status. We derive a number of criteria from different neighborhood-based outlier detection techniques. With the exception of one technique, our approaches have low complexity and running times. In our experiments, we compare against two recently proposed techniques from the field that are have more sophisticated theoretical foundations, as well as against two well-established kNN classifiers. We find that our approaches are competitive with existing work and especially that the recent techniques do not constitute an improvement.

CR Subject Classification : I.2, H.2.8

Distance-based adaptive k-neighborhood selection

Albrecht Zimmermann
albrecht.zimmermann@cs.kuleuven.be

KU Leuven, Celestijnenlaan 200A, Leuven, B-3001 Belgium

Abstract. The k-nearest neighbor classifier follows a simple, yet powerful algorithm: collect the k data points closest to an unlabeled instance, according to a given distance measure, and use them to predict that instance's label. The two components, the parameter k governing the size of used neighborhood, and the distance measure, essentially determine success or failure of the classifier. In this work, we propose to reverse the use of outlier-detection techniques that are based on k-neighborhoods in order to determine the value of k. To achieve this, we invert the workings of these techniques: instead of using a fixed k to decide whether an instance is an outlier, we stop growing the k-neighborhood as soon as the unlabeled instance would be given outlier status. We derive a number of criteria from different neighborhood-based outlier detection techniques. With the exception of one technique, our approaches have low complexity and running times. In our experiments, we compare against two recently proposed techniques from the field that are have more sophisticated theoretical foundations, as well as against two well-established kNN classifiers. We find that our approaches are competitive with existing work and especially that the recent techniques do not constitute an improvement.

1 Introduction

k-nearest-neighbor (k-NN) classification [10] follows a simple schema: given an unlabeled instance, find the k nearest labeled instances according to some distance measure, and combine their labels into a prediction. This so-called "lazy" learning approach is very efficient during training, albeit at the cost of potentially expensive search for the nearest neighbors during prediction, and despite its apparent simplicity, one can give strong guarantees for its error rate. Specifically, [4] proved that 1-NN has asymptotically at worst twice the Bayes error rate, and that the Bayes error can be approached further – for $k \rightarrow \infty$ and $\frac{k}{n} \rightarrow 0$, with n the cardinality of the data. This is a significant guarantee in that the Bayes error rate can effectively be considered the minimum error rate attainable on the data.

In most settings, only a limited amount of data is available, however and the optimal value of k is different for different data sets. The remaining problem is how to decide on that value of k. An empirical way consists of using a validation

set to determine this value during training, unfortunately importing the long prediction search times into the training phase.

Choosing the right k is arguably less difficult for *typical* instances, i.e. instances that are representative of the generating process of the class and have therefore many similar neighbors of that class. A small k will pick from a cluster of instances representing the same class, while a large k is not in danger of diluting the prediction with instances that are not members of the class. *Atypical* instances, on the other hand, i.e. outliers within the class, possibly ones that lie at class boundaries, can be expected to be in a situation where, depending on the k, instances from several classes are included.

This problem has been realized before and given rise to different approaches towards addressing it, such as discounting the influence of farther neighbors by assigning each instance a weight inversely proportional to its distance from the unlabeled instance. Depending on the interplay between distances and a (fixed) k, this can still have the effect of one close neighbor being overruled by several more distant ones. Other approaches adjust the distance metric.

One way of looking at instances that are atypical w.r.t. their class is as *outliers*, and outlier detection is another field that has to deal with the question of distances between neighboring points and whether to let them influence an instance's label. Generally speaking, an outlier is an instance that is sufficiently different from other instances near it to be generated by a different generating process. One way of assessing this difference lies in comparing it to the difference any two other points in the data have – at which point the problem appears that compared to, e.g. the distance of *other* outliers from the main body of the data, an outlier might appear "normal".

Interestingly enough, there exist outlier detection techniques based on k-neighborhood concepts, which label an instance an outlier if it deviates too much from its local neighborhood. Intuitively, these techniques' test takes the form: "Given k, is the instance similar (enough) to its k-neighborhood?". If it is not, it is labeled an outlier. In this work, we propose to turn this relationship around: instead of letting the k-neighborhood decide whether an instance is an outlier, we let the instance decide what is an appropriate neighborhood. The test becomes: "Given the instance, how far can we increase k before the instance is not similar to its k-neighborhood anymore?" A number of different approaches towards identifying the degree of outlierdom of an instance have been proposed and we evaluate them w.r.t. their applicability for choosing the right neighborhood. In particular, this means that not all unlabeled instances will have their label predicted by the same number of nearest neighbors.

The contributions of this work are as follows:

1. We illustrate how to reverse engineer outlier detection techniques to adjust the k parameter based on characteristics of the testing instance and the training data.
2. We derive several novel adaptive kNN techniques based on this principle.
3. We show experimentally that the proposed techniques are competitive with existing techniques. Furthermore, we show that recently proposed classifiers

based on a more sophisticated theoretical background *fail* to improve on our simpler techniques or established kNN methods.

In the following section, we quickly clarify our concept of atypical instances to give some intuition of why the k-NN classifier can run into problems for certain class distributions. In section 3, we discuss existing attempts to address this problem, either by modifying the distance metric or adaptively choosing an effective k. In Section 4, we discuss the work in k-neighborhood based outlier detection and reverse engineer them in Section 5 to propose formulations of these techniques that help determine k for an unlabeled instance. We evaluate different parameter settings for these newly proposed techniques and compare them against existing approaches in Section 6, before we conclude in Section 7.

2 (A)typical Instances

As an illustration of the problem, consider Figure 1. Not only would a seemingly clear-cut labeling become murkier if k went from two to three but increasing k further would give more weight to the right-hand cluster of points. In addition, the two nearest neighbors are arguably less typical for "their" cluster than the third-closest neighbor is for its cluster. This means in turn both n_1 and n_2 might not be classified correctly themselves as soon as $k > 1$.

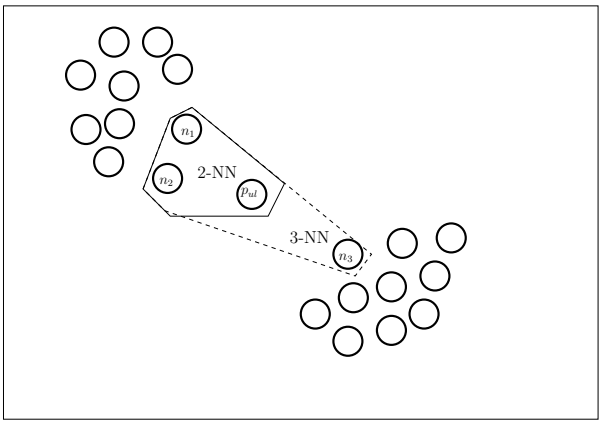

Fig. 1. Effects of changing k on the involved k-neighborhood

There are other atypical instances, e.g. ones that lie outside a class cluster but not close to a class boundary or those that are surrounded by instances of another class. We are not concerned with these cases, however, since in the first case, different k will rarely lead to a mis-classification, while in the second case different k will probably not lead to a correct classification.

3 Related Work

The original k-NN technique was proposed more than half a century ago [10], and a large amount of work has been done on speeding up the search for nearest neighbors of instances to be labeled, the bottleneck of the technique.

At the same time, however, deciding on the proper value for k, and using the derived k-neighborhood effectively has been an open topic from early on as well. Cover *et al.* [4] proved that the 1-NN rule is asymptotically at worst twice as bad as Bayes error, as well as that k-NN approaches the Bayes error for $k \to \infty, \frac{k}{n} \to 0$. Unfortunately, there is no clear-cut solution as to how to select k for limited n, a task that is further complicated by the fact that k-NN is strongly affected by the "curse of dimensionality" – i.e. that for high-dimensional data all instances seem to be at the same distance. In fact, one of the earliest works addressing this issue is Dudani's paper from 1975 [8], in which neighbors' influence is weighed in indirect proportion to the distance from the instance to be labeled.

More generally, this topic has given rise to two classes of "adaptive" k-NN methods: 1) those that learn or adjust the distance metric, and 2) those that dynamically adjust k, depending on characteristics of the training data and/or the unlabeled instance.

The former includes [14] in which the author uses LDA to identify decision boundaries and shrinks neighborhoods orthogonally to them in the DANN algorithm, [6, 7] in which an SVM is used to identify the "most discriminant direction" over the unlabeled instance's neighborhood which is then used to weigh features, and [19] in which the "most informative" points are identified by use of a class-distribution dependent metric and then used for labeling.

[20] also present their work in terms of adjusting the distance metric but theirs is a special case. First, this paper proposes to turn the k nearest neighbors into a series of centroids consisting of successively more points, to model distributional changes. In addition, the distance of the unlabeled instance to those k centroids consists of a *normalized* term that rewards close centroids further and discounts farther ones, and a second term that rewards low entropy of class labels among the points involved in a centroid.

The latter category includes [15] which proposes turning k-NN into a probabilistic classifier (PNN) by marginalization over k, making it independent from k and leading to smoother decision boundaries, and [12] that uses Bayesian likelihood estimates more directly to decide on the value of k. The baseline all this techniques attempt to improve upon consists of determining the k to be used by cross-validation over the training data, a method that will lead to the same k being used for all unlabeled points.

[13], finally, does both – adjust the distance metric and select the number of nearest neighbors to use for labeling the new instance by combining the ideas of the DANN and PNN algorithms.

The focus of our approach lies in exploiting distance information to select a good number of neighbors, i.e. it belongs in the second category. We abstract

from the type of distance metric used, and also do not exploit class label information, differing from the all the other works described in this section.

4 Outlier Detection

Outliers are instances in the data that do not seem to be generated by the same processes as other data points. A number of different techniques towards solving this problem have been developed (cf. [2]). One of the dimensions along which approaches can be distinguished is whether they are supervised or unsupervised, i.e. whether "normal" instances are labeled as such for building an "outlier detector" or not. Another point of view distinguishes outlier detection methods between those that build an explicit model of "normal" and those that use an implicit characterization.

4.1 kNN-based outlier detection

k-nearest neighbor outlier detection methods fall into the second category in both cases: all instances are considered unlabeled and instead of an explicit model, "typical" distances are used to identify "atypical" ones. Given a fixed k-neighborhood, an instance's distance information can be used in different ways to decide that it is an outlier:

1. The instance is more than $dist_{max}$ from its closest neighbor [16].
2. The *sum* of distances to its k nearest neighbors exceeds $dist_{max}$ [9].
3. At most λ points lie within distance $dist_{max}$ from the instance [17].
4. The distance to the kth-nearest neighbor is smaller than the distance to no more than λ other (non-neighborhood) points [18].
5. The average distance of the instance to the points in its k-neighborhood is comparatively larger than the average distance of all points in the k-neighborhood to points of *their* k-neighborhood, as expressed by the *local outlier factor* (lof) [1].

5 Adaptive k-determination

With the exception of the first, all the techniques described in the preceding section can be interpreted directly in terms of limiting k in such a way that the unlabeled instance will *not* be an outlier with regard to the k-neighborhood. We denote the unlabeled instance by d_{ul}, the data set by $\mathcal{D}$, the distance metric by $d : \mathcal{D} \times \mathcal{D} \mapsto \mathbb{R}^+$, and $\lambda, k \in \mathbb{N}$, $dist_{max} \in \mathbb{R}^+$. In the following, we revert the workings of the outlier detection techniques to derive methods for adapting k to the characteristics of the unlabeled instance and the training data, and give interpretations of the criteria:

1. No mapping.

2. SumDist: Select

$$k = \arg\max_k \left\{ \left(\sum_{p \in k-neighborhood} d(p_{ul}, p) \right) \leqslant dist_{max} \right\}$$

This criterion can be expected to address situations such as the one shown in Figure 1 in that only few "nearest" neighbors would be involved for instances whose nearest neighbors are at a large distance. If, on the other hand, p_{ul} lies in a dense region, near neighbors will crowd out farther away ones.

3. MaxDist: Select $k = |\{p \mid d(p_{ul}, p) \leqslant dist_{max}\}|$ This selection criterion effectively replaces the choice of k with a choice for a maximum distance that should not be exceeded by any point involved in labeling. The motivation for such a criterion is straight-forward since we would expect far away neighbors not to be very informative w.r.t. p_{ul} even if they are comparatively "close".

4. NotFarthest: Select k such that

$$|\{p \mid d(p_{ul}, p) \geqslant \max_{p_n \in k-neighborhood} d(p_{ul}, p_n)\}| \geqslant \lambda$$

This selection criterion can be expected to be more flexible than criterion 2. since it trades off maximum distance within p_{ul}'s neighborhood against its distances to the rest of the data. Instead of requiring that neighbors be within a given distance, the requirement changes into being "nearer than λ others". In a sense, the classifier is changed from a nearest-neighbor classifier to a "not-farthest neighbor" one.

5. LofNeighborhood: Select k such that the average distance of p_{ul} to the points in its k-neighborhood is not significantly larger than the average distance of those points to *their* k-neighborhoods:

$$\arg_k \max lof_k(p_{ul}) \leqslant \theta$$

The local outlier factor effectively compares the density of p_{ul}'s k-neighborhood against the density of its neighbors' k-neighborhoods and only selects those neighbors that show similar density characteristics (and can therefore be expected to be generated by the same process).

Additionally, we propose a relaxation of 3.:

6. λ-MaxDist: Select k such that

$$|\{p \in k - neighborhood \mid d(p_{ul}, p) \geqslant dist_{max}\}| < \lambda$$

In each case, the classification is performed at least with $k = 1$.

It should be noted that, with the exception of LofNeighborhood, our proposed techniques can be expected to be computationally *less* expensive than selecting k based on a leave-one-out (loo) cross-validation, a well-established kNN approach: while the latter needs to evaluate a range of k values on all training data, our approaches evaluate only for testing data and have clear stopping criteria. Furthermore, our techniques base on a *less* complex theoretical background and *fewer* assumptions than the alternative techniques we have discussed in Section 3.

6 Experiments

To evaluate the effectiveness of the proposed techniques, we performed classification experiments on a variety of UCI data sets, listed in Table 1. The table lists the size of the data set, its dimensionality, the number of classes, and whether the class distribution is skewed. We labeled a data set as skewed if the size of any two classes in the data differs by a factor of two or more. For data sets with nominal attributes, we performed binarization of the attributes and used Manhattan distance, for data sets with numerical values Euclidean distance. Classification accuracy was estimated performing stratified 10-fold cross-validation. We compare to the standard k-nearest neighbor classifier, and the distance-weighted k-nearest neighbor, as implemented in the WEKA tool kit [11].

Data sets with nominal attributes					Data sets with numerical attributes				
Name	Size	Dim.	Classes	skewed?	Name	Size	Dim.	Classes	skewed?
Audiology	226	69	24	yes	Breast Cancer (Wisc.)	699	9	2	yes
Breast Cancer	286	9	2	yes	Diabetes (Pima)	768	8	2	yes
Car	1728	6	4	yes	Ecoli	336	7	8	yes
Dermatology	366	34	6	yes	Glass	214	9	6	yes
Kr vs Kp	3196	36	2	no	Heart Statlog	270	13	2	no
Lung Cancer	32	57	3	no	Ionosphere	351	34	2	yes
Lymphography	148	18	4	yes	Iris	150	4	3	no
Mol. Biology - Prom.	106	58	2	no	Liver Disorders	345	6	2	no
Post. Patient Data	90	8	3	yes	Page Block	5473	10	5	yes
Primary Tumor	339	17	21	yes	Sonar	208	60	2	no
Soybean	683	35	19	yes	Spectrometer	531	102	48	yes
Splice Junction	3190	61	3	yes	Vehicle	846	18	4	no
Tic Tac Toe	958	9	2	yes					
Voting Record	435	16	2	yes					

Table 1. Characteristics of data sets used in the experimental evaluation

In addition, we compare against the techniques described recently in [12] and [20], using the standard parameters proposed in those works. An implementation of the former was provided to us by the author. We reimplemented the latter technique after consultation with the authors, and introduced a slight change: the authors use a distance measure of the form:

$$d_c = c_1 + \alpha \cdot c_2,$$

with α a balance parameter. We changed this formulation to the more common:

$$d_c = \alpha \cdot c_1 + (1 - \alpha) \cdot c_2.$$

Both of those techniques were shown in the referenced publications to significantly outperform standard kNN with k selection via internal loo cross validation.

Given the relatively low complexity and running times of our techniques, we would therefore consider our proposal successful if we approach the accuracy of those two approaches.

We propose a total of five different techniques and aim to compare those against the four existing approaches. Since we want to evaluate different parameter settings for our approaches to gain an understanding how settings influence classification accuracy, we quickly run up against a limit in informativeness that 26 data sets can provide. In the following subsections, we therefore evaluate different parameter settings for each proposed approach and attempt to link them to characteristics of the data. Since it is difficult to map data set characteristics to performance by hand, we encoded all data sets in terms of the information given in Table 1, as well as normalized distances for the first, second, fourth decile, as well as minimum and median distance, and used WEKA's J48 to try and learn a model mapping the characteristics to the best-performing algorithm.

6.1 SumDist parameter evaluation

For this technique, we evaluate the median, and maximum distance in the data, and find no significant difference for either setting. Using the median distance performs better than using the maximum distance on ten data sets, on fifteen data sets this is reversed and on one data set both methods attain the same classification accuracy. It is however difficult to characterize when either method can be expected to perform well. Given that the parameter is linked to the distribution of distances in the data, we would have expected to see this reflected but as Figure 2 shows, this is not the case. The figure shows the median distance normalized by the maximum distance, and while the result of the shared accuracy seems intuitive – a median distance almost as high as the maximal one – there is no clear trend for the other two blocks. The situation does not become clearer when data set characteristics are considered. J48 achieves 88% accuracy (3 misclassified data sets). The tree first tests whether number of attributes is $\leqslant 57$ before skewness and other characteristics come into play.

6.2 MaxDist parameter evaluation

For this technique, we evaluate the use of the median distance in the training data as $dist_{max}$, as well as first, second, fourth decile distance. Once again, while there is one setting that performs better than the rest (using the first decile distance), it is best on only slightly more than half of the data sets. Figure 3 shows how as in the case of the SumDist approach, there is no clear trend discernible. In particular, for the first decile, the distance ranges from very low to almost half the maximum distance. Also, as for the preceding approach, data sets show no trends for characteristics either. J48 achieves 76.9% accuracy (6 misclassifications), and the decision tree consists only of cardinality tests, somewhat surprising given that the method's parameter is distance-based.

6.3 NotFarthest parameter evaluation

For this technique, we evaluate λ equal to 70%, 80%, and 90% of the training data. Our implementations were unfortunately rather inefficient so that the experiments for the Page Blocks data set had not finished at the time of writing.

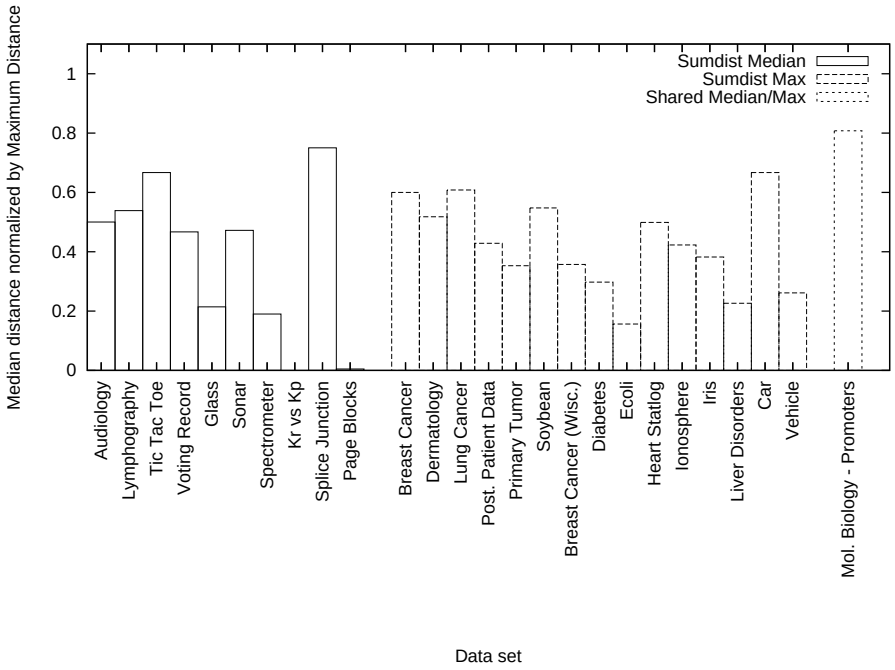

Fig. 2. Performance of SumDist settings in relation to relative median distance values

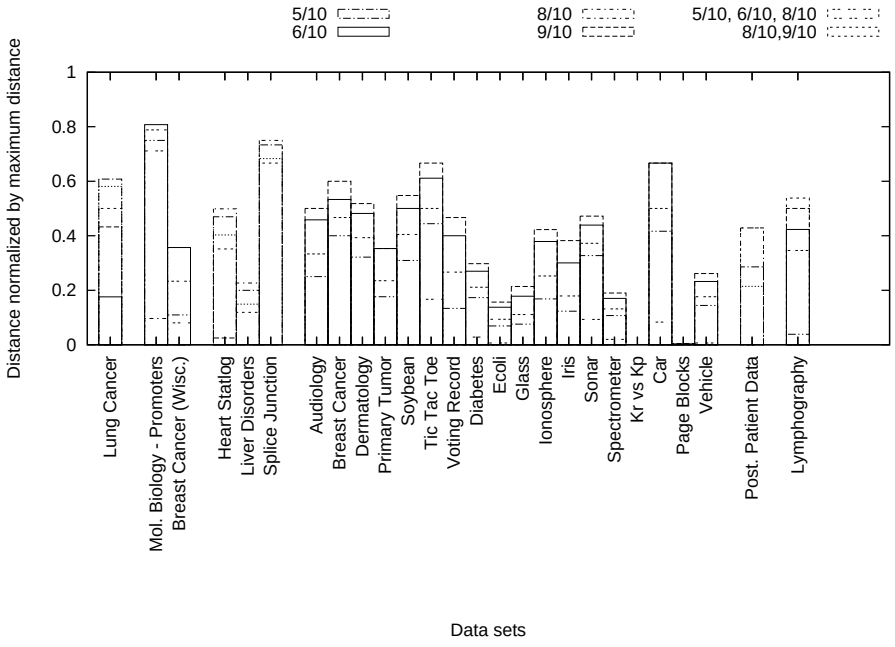

Fig. 3. Performance of MaxDist settings in relation to relative decile distance values

Using 90% of the data was best on twelve data sets, 80% on nine, and 70% on four. These results prove hard to model for J48, which achieves only 61.5% accuracy. The decision tree relates entirely to classes, their number and skewness. This might mean that the λ parameter is dependent on the class that unlabeled instance belongs to but since this information will not be known, this is not helpful.

6.4 LofNeighborhood parameter evaluation

For this technique, we evaluate lof=1 and lof=1.2. Since for instances that lie inside very dense regions, lof can always be below 1 – they are very clearly *not* outliers – we stop adding points to the neighborhood if the prediction for the label has not changed for a number of additional points equal to 5%/10% of the data. This is by far the most expensive classifier since adding points to the k-neighborhood results in having to compute new neighborhoods and recomputing existing ones. As a result of this, the running times for Kr vs Kp, Splice Junction, and Page Blocks were in excess of twenty-four hours per fold and we did not perform experiments on these data sets for LofNeighborhood.

LofNeighborhood differs from the other techniques in that there is no single setting that performs best clearly more often than other. Instead, the highest number of data sets on which a single setting is best is six (lof=1.2, 5%). J48 achieves 69.2% accuracy and the tree involves cardinality tests close to the root.

6.5 λ-MaxDist parameter evaluation

Finally, for this technique we evaluate the same distances as for MaxDist and λ equal to 5%, 10% of the training data. Once again, however, there is no trend to the distances of the data sets in relation to distance parameter settings, as shown in Figure 4. The only parameter setting which has a clear impact is using the first decile distance and $\lambda = 5$ which performs best on fifteen data sets and ties for best on five others. J48 achieves 84.6% accuracy, and the tree's tests are mainly on cardinality and number of attributes.

6.6 Choosing parameters for existing techniques

We ran the technique proposed in [12] with the settings described in the paper, with one exception. The author argues that without any additional prior information a uniform prior is the best choice but since we perform a stratified cross-validation, we do know that the empirical class distribution in the training data reflects the true class distribution and therefore evaluate this prior as well. For this technique, we used two different error measures: 1) the one used in the implementation, which weighs classification error by class priors, in effect discounting classification mistakes made on minority classes, and 2) an unweighted one, which corresponds to the typical error measure in dividing the number of

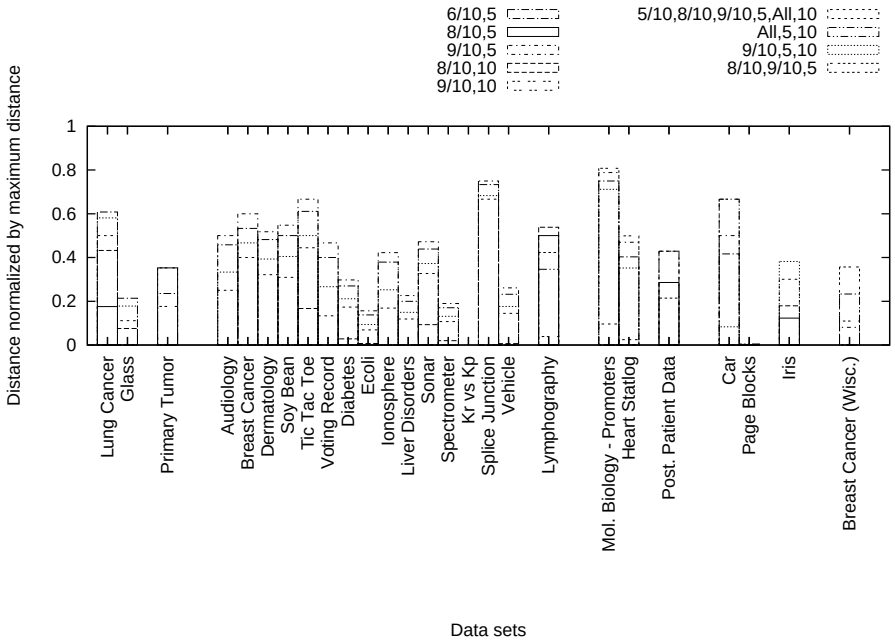

Fig. 4. Performance of λ-MaxDist settings in relation to relative decile distance values

misclassifications by the total number of test instances. The empirical prior is more successful, winning on 12 data sets and tying on 7. J48 achieves 84.6% accuracy, the tree tests number of classes first but there is not easily describable concept. While the number of classes does not seem to have an effect on the usefulness of the prior, having more than two classes favors the discounted error estimate.

For the centroid-based technique proposed in [20], we evaluate $\alpha \in \{0.25, 0.5, 0.75\}$. Since the authors show stability w.r.t. k in their experiments, we evaluated $k \in \{3, 5\}$. Furthermore, the authors propose both classification by the nearest centroid and by majority voting of all centroids. We find that when using the majority vote and $k = 5$, α does not have an influence on the accuracy: the three settings tie for best on thirteen data sets. When using the nearest centroid classification, however, $\alpha = 0.25$ leads to the best result six times for $k = 5$ and four times for $k = 3$. Generally, we cannot reproduce the authors' claim that their technique is stable w.r.t. k. J48 achieves 73.1% accuracy, tests first minimum distance, and then focusses on class information.

6.7 Evaluating different techniques

As the experiments in the preceding sections showed, the decision how to set parameters for kNN variants, both the ones we propose in this paper, and the ones proposed before, is not an easy one to make. In many cases there is no

clearly superior setting, in line with the "no free lunch" assumption in machine learning. In addition, the information about the data set characteristics seems not to be enough to reliably decide on settings. We have used the experiments to identify for each data set and technique which parameters work best, and while no setting managed to distinguish itself, things might be different when comparing different approaches.

In the final evaluation, we compare the best settings for each of our proposed techniques, and the two existing approaches against k-NN and distance-weighted k-NN as implemented in the WEKA toolbox. In the latter two cases, k is selected via internal cross-validation.

Given the low complexity of our proposed approaches, it comes as a positive surprise that they perform competitive both to the computationally more expensive standard kNN with internal loo cross validation, and to the more complex methods that have been proposed recently. In fact, applying the Friedman/Nemenyi procedure [5], we fail to find any significant differences at all (see Table 2)! With the exception of LofNeighborhood each technique is best- or second-best ranked at least once, while the technique with the best average rank is the basic k-NN with k selected by internal cross-validation.

Technique	Average Rank
SumDist	5.0384
MaxDist	5.0961
NotFarthest	4.3846
LofNeighborhood	6.1538
λ-DistMax	5.94
c-NN	4.5
Bayesian Adaptive k-NN	4.8077
Distance-weighted k-NN	4.673
k-NN	4.4038

Table 2. Overall comparison of different k-NN interpretations – critical distance 2.16 ($p = 0.1$)

This is also somewhat surprising, given that both [12] and [20] report *significant* improvements over cross-validated k-NN in experiments on UCI data sets, results we cannot reproduce. Our finding does, however, agree with the "no free lunch" assumption, i.e. that there is no single algorithm that can be expected to outperform all others on all data. Unfortunately, at this point, we are not able to identify a clear relationship between data set characteristics and good performance of particular interpretations of the k-NN framework.

7 Summary and Conclusion

In this work, we have leveraged k-neighborhood based outlier detection techniques for adapting k to the characteristics of an instance to be labeled by a

k-NN classifier. We discussed a number of such outlier detection techniques, and showed how they can be reverse engineered to choose k based on distance information relating the unlabeled instance and the available training data.

We evaluated a number of parameter settings and compared the proposed techniques against existing work in adaptive k-NN, as well as the baseline of choosing k by internal cross-validation. The proposed methods are competitive with existing techniques while at the same time having lower computational complexity and making fewer assumptions regarding the data. We also find that none of the existing works has a clear advantage over our newly proposed techniques, or other existing approaches. This is remarkable given the age of the original k-NN classifier and the large body of work that has attempted to improve on it.

It would be tempting for us to declare that since all k-NN interpretations seem to perform similar, one should just choose the fastest one, i.e. one of ours. We believe that it should be possible to do better, however. On the one hand, we made the conscious decision to use only basic Euclidean and Manhattan distance in our experiments, and the solution might after all be found in adjusting the distance metric. On the other hand, while there might not be free lunch, identifying under which characteristics which approaches to k-NN classification succeed (or fail) is in our opinion an important question of machine learning research that we intend to explore in the future.

References

1. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: Identifying density-based local outliers. In: Chen et al. [3], pp. 93–104
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. 41(3) (2009)
3. Chen, W., Naughton, J.F., Bernstein, P.A. (eds.): Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA. ACM (2000)
4. Cover, T., Hart, P.: Nearest neighbor pattern classification. Information Theory, IEEE Transactions on 13(1), 21 – 27 (Jan 1967)
5. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
6. Domeniconi, C., Gunopulos, D.: Adaptive nearest neighbor classification using support vector machines. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) NIPS. pp. 665–672. MIT Press (2001)
7. Domeniconi, C., Gunopulos, D.: Efficient local flexible nearest neighbor classification. In: Grossman, R.L., Han, J., Kumar, V., Mannila, H., Motwani, R. (eds.) SDM. SIAM (2002)
8. Dudani, S.: The distance-weighted k-nearest-neighbour rule. IEEE Transactions on Systems, Man and Cybernetics SMC-6(4), 325–327 (1975)
9. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In: Applications of Data Mining in Computer Security. Kluwer (2002)
10. Fix, E., Hodges, J.L.: Discriminatory analysis, nonparametric discrimination: Consistency properties. US Air Force School of Aviation Medicine 4(3), 477 (January 1951)

11. Frank, E., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)
12. Ghosh, A.K.: On optimum choice of k. Computational Statistics & Data Analysis 50(11), 3113–3123 (2006)
13. Guo, R., Chakraborty, S.: Bayesian adaptive nearest neighbor. Statistical Analysis and Data Mining 3(2), 92–105 (2010)
14. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification. IEEE Trans. Pattern Anal. Mach. Intell. 18(6), 607–616 (1996)
15. Holmes, C.C., Adams, N.M.: A probabilistic nearest neighbour method for statistical pattern recognition. Journal of the Royal Statistical Society - Series B: Statistical Methodology 64(2), 295–306 (2002), http://doi.wiley.com/10.1111/1467-9868.00338
16. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: Gupta, A., Shmueli, O., Widom, J. (eds.) VLDB. pp. 392–403. Morgan Kaufmann (1998)
17. Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-based outliers: Algorithms and applications. VLDB J. 8(3-4), 237–253 (2000)
18. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: Chen et al. [3], pp. 427–438
19. Song, Y., Huang, J., Zhou, D., Zha, H., Giles, C.L.: Iknn: Informative k-nearest neighbor pattern classification. In: Kok, J.N., Koronacki, J., de Mántaras, R.L., Matwin, S., Mladenic, D., Skowron, A. (eds.) PKDD. Lecture Notes in Computer Science, vol. 4702, pp. 248–264. Springer (2007)
20. Zhang, Q., Sun, S.: A centroid k-nearest neighbor method. In: Proceedings of the 6th international conference on Advanced data mining and applications: Part I. pp. 278–285. ADMA'10, Springer-Verlag, Berlin, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1947599.1947627